COM S 476/576 Project 4: Motion Planning for a Dubins Car

Due Apr 21 at 11:59pm

In this project, you will develop a path planning algorithm for systems with differential constraints, specifically a car-like system.

Dubins car: Recall that the configuration of a simple car is given by $q = (x, y, \theta)$, where (x, y) is the position of the center of its rear axle and θ is its orientation. Let s and ϕ denote the speed and the steering angle (negative when the front wheel is turned to the right and positive when it is turned to the left). See Figure 1, taken from the textbook. The motion of the car can be described by the following configuration transition equation

$$\begin{aligned}\dot{x} &= s\cos\theta \\ \dot{y} &= s\sin\theta \\ \dot{\theta} &= \frac{s}{L}\tan\phi\end{aligned}$$

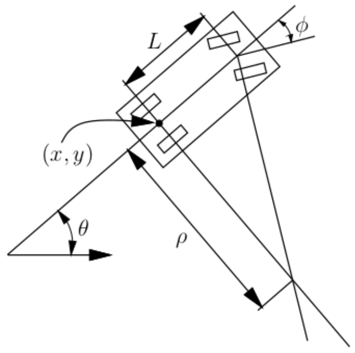

Figure 1: The simple car with three degrees of freedom.

Consider the Dubins version of the simple car. We assume that the car moves at constant speed $s = 1$ and $\phi \in [-\phi_{max}, \phi_{max}]$, where $\phi_{max} \in (0, \pi/2)$. The minimum turning radius is $\rho_{min} = L/\tan\phi_{max} = 0.5$. It can be shown that the shortest path between any two configurations can always be characterized by a Dubins curve, which is one of the following types: LRL, RLR, LSL, LSR, RSL, RSR. Here, L and R corresponds to turning as sharply as possible to the left and right, respectively, while S corresponds to driving the car straight.

The world and obstacle region: The car is operating in a 2D world $\mathcal{W} = [-3, 3] \times [-1, 1]$ with obstacle region $\mathcal{O}$ shown in Figure 2. The obstacle region is defined by two half circles centered at $(0, -1)$ and $(0, 1)$, respectively, both having radius $1 - dt$, where $dt = 0.2$. Anything within the two half circles are considered obstacle regions. (This is similar to Project 3 except that Project 3 defines the C-space rather than the world and dt is larger in this project.)

The initial configuration is $q_I = (-2, -0.5, 0)$ and the goal configuration is $q_G = (2, -0.5, \pi/2)$.

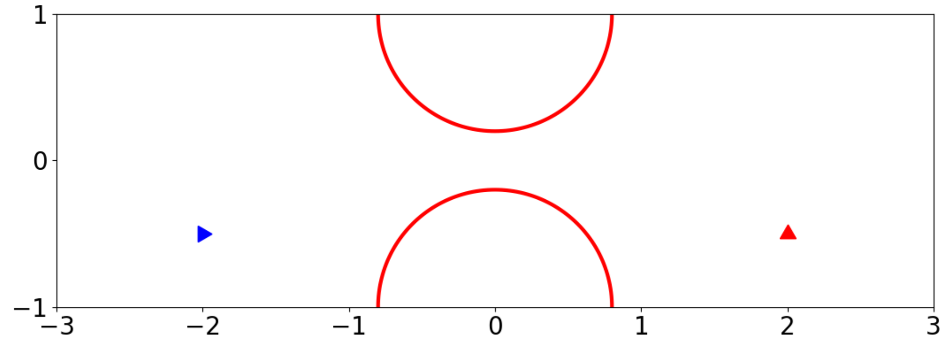

Figure 2: The world and obstacles. The red half-circles are the obstacles. The blue and red arrow represent the initial and goal configurations, respectively.

Task 1 (Solve the planning problem using RRT) [20 points for 476, 10 points for 576]: Use the single-tree search outlined in Section 14.4.3. You should check periodically if the tree can be connected to q_G. This can be easily done by setting $\alpha(i)$ as q_G with a certain probability p. For example, the book recommends $p = 0.01$. You should have this as a parameter in your code. Once q_G is successfully added to the tree, quit the loop and compute the path from q_I to q_G. Plot both the resulting tree and path. The result should be similar to Figure 3, which uses $p = 0.1$.

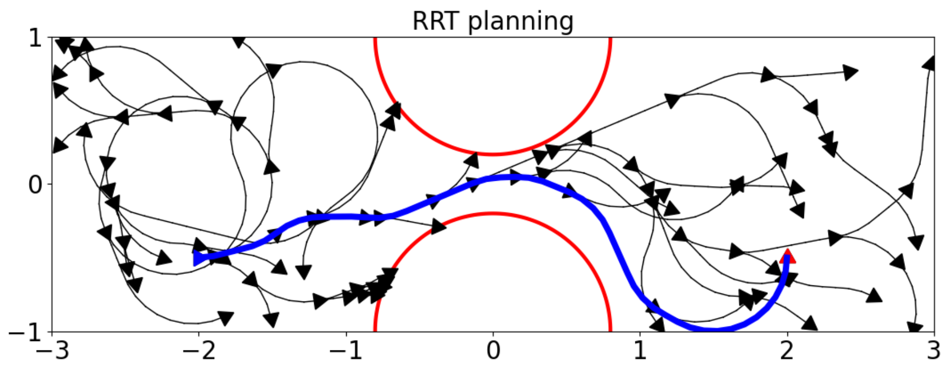

Figure 3: The result of RRT planning with p = 0.1. The black curves are the edges in the tree. The black arrows are the configurations of the vertices. The blue curve is the path from q_I to q_G extracted by RRT.

Hint:

1. **Sampling:** As opposed to Project 3, a configuration in this project includes 3 values x, y and θ. Make sure you provide an appropriate range for all of them.

2. **Dubins curves:** There are existing libraries for computing the shortest paths between any configurations for a Dubins car. Feel free to use any of them. For example, if your code is in python, you may consider the `dubins` library (`https://pypi.org/project/dubins/`). OMPL also includes the `ompl::base::DubinsStateSpace` class (`https://ompl.kavrakilab.org/classompl_1_1base_1_1DubinsStateSpace.html`), which allows you to compute the shortest Dubins path between $SE(2)$ configurations.

3. **Collision checking:** Besides checking for collisions with the 2 half-circle obstacles, for this project, you also need to check that each Dubins curve remains within the world $\mathcal{W} = [-3, 3] \times [-1, 1]$. In Project 3, this check is not necessary because an edge between any 2 configurations are simply a straight line and the configuration space is convex, so as long as all the samples are within the configuration space, any line segment between them is also within the configuration space.

4. **Approximate solutions for finding nearest points:** For each edge, discretize the corresponding Dubins curve such that consecutive points are not further than Δq from each other. When finding the nearest point in the swath S of $\mathcal{G}$ to $\alpha(i)$, you can simply go through these discretized states and select one that is closest to $\alpha(i)$.

Task 2 [required for 576, bonus for 476] (Solve the planning problem using PRM) [10 points for 576, 5 bonus points for 476]: This last part of the assignment is required for those taking COM S 576 only. 5 bonus points will be given to those taking COM S 476 who correctly completes the task.

Please use the PRM algorithm described in Figure 5.25 in the textbook to solve the same planning problem given above. As for the number of nodes N, try a different value until you can solve the problem. When connecting to the nodes in the neighborhood, use Nearest K and set $K = 15$.

Hint: As opposed to Project 3, in this project, an edge from configuration q_2 to configuration q_1 may not simply be the reverse of an edge from q_1 to q_2. This is because a Dubins car can only move forward so it cannot simply traverse an edge from q_1 to q_2 backward to obtain an edge from q_2 to q_1. The `CONNECT` function for PRM, therefore, needs to perform collision checking for both the forward and reverse edges.

Submission: Please submit a single zip file on Canvas containing the followings

- your code (with comments, explaining clearly what each function/class is doing),
- the plots from each task, similar to Figure 3, and
- a text file explaining clearly how to compile and run your code.